

Re: recvfrom() strange operation

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.system/2006-09/msg00178.html>

- *From:* kzsolt@xxxxxxxxxxx
 - *Date:* 19 Sep 2006 01:27:54 -0700
-

First sorry for the long delay but from our country (hu) the sign in to google is impossible over more days (reason: page not found). I'm using some tricky links to sign in. This I hope will work for a long time.

"You must always use the upper layer protocol to handle duplicated or mis-ordered packets when using UDP, no matter what your network looks like."

That is what I do. Our sftp has framing and retransmission capability, discovers packet loss, partially lost and joined packets and octets reverse too. And the most important sftp works fine over TCP or UDP (the first ~30% slower).

Edifying result: Triple buffering is more than 70% efficient than double buffering!!! The reason (maybe) the `recv()` or `send()` system call costs much resources.

steve_schefter@xxxxxxxxxxx wrote:

kzsolt@xxxxxxxxxxx wrote:

"3) Have the ..."

But in that case not working, because the framing always requires buffer space (for escape sequence).

This does not preclude avoiding a copy. You just have to handle "frames" (or whatever you are calling your higher layer protocol element) crossing UDP message boundaries.

Or better yet, put some knowledge of the fact that you are using UDP into the code and your "frame" will exist entirely within one datagram (assuming it will fit). You then won't have to have escape characters to mark your frame boundaries.

Other side our task is "using with standard calls", this means we cannot go deeply to kernel, and we can migrate any time to x86-solaris.

Re: recvfrom() strange operation

Clearly I'm not following something here. Earlier you complained about having to write the same algorithm in both kernel and user space and I therefore noted that you don't need to get into user space just to use sockets. Now you tell me that you want portability to x86solaris which would mean that you need to be doing it all in user space since there's no common DDI/DKI in both Linux and Solaris. So how does the need to have the algorithm in kernel space (and thus the duplication) come into it?

This is maybe the only reason why UDP do not enable to read packet as octett only as packet.
But "application" can (must) discover packet squence reverse from octett stream, so this is pain for us....

We can only guess what was in the mind of the writers of IPv4, but I'd be more inclined to guess that you can only read UDP as packet because it is a packet-based while you can read TCP as bytes because it is byte-based (streaming). It still sounds to me like you are trying to use UDP as a streaming protocol, which it is not.

Other sides the UDP packet sequence problem littlebit mistificated. If you have complex IP network (maybe generate packet squence reverse) then you must use TCP instead of UDP. Otherwords in this kind of network UDP has more disadvantege then advantage.

If you are relying upon the network to deliver UDP in order even for the simplest network, then you have a serious design flaw. You must always use the upper layer protocol to handle duplicated or mis-ordered packets when using UDP, no matter what your network looks like.

Regards,
Steve